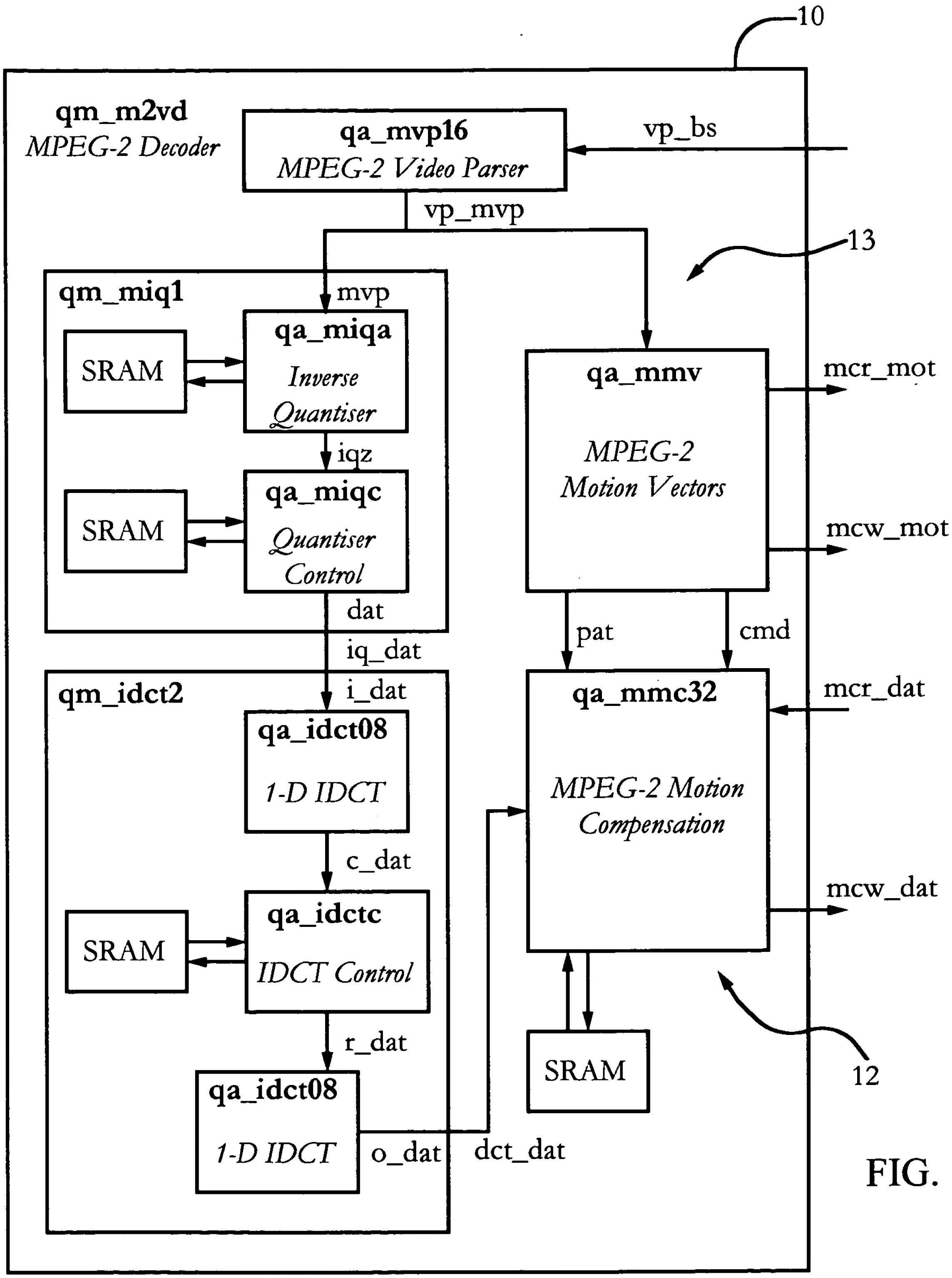
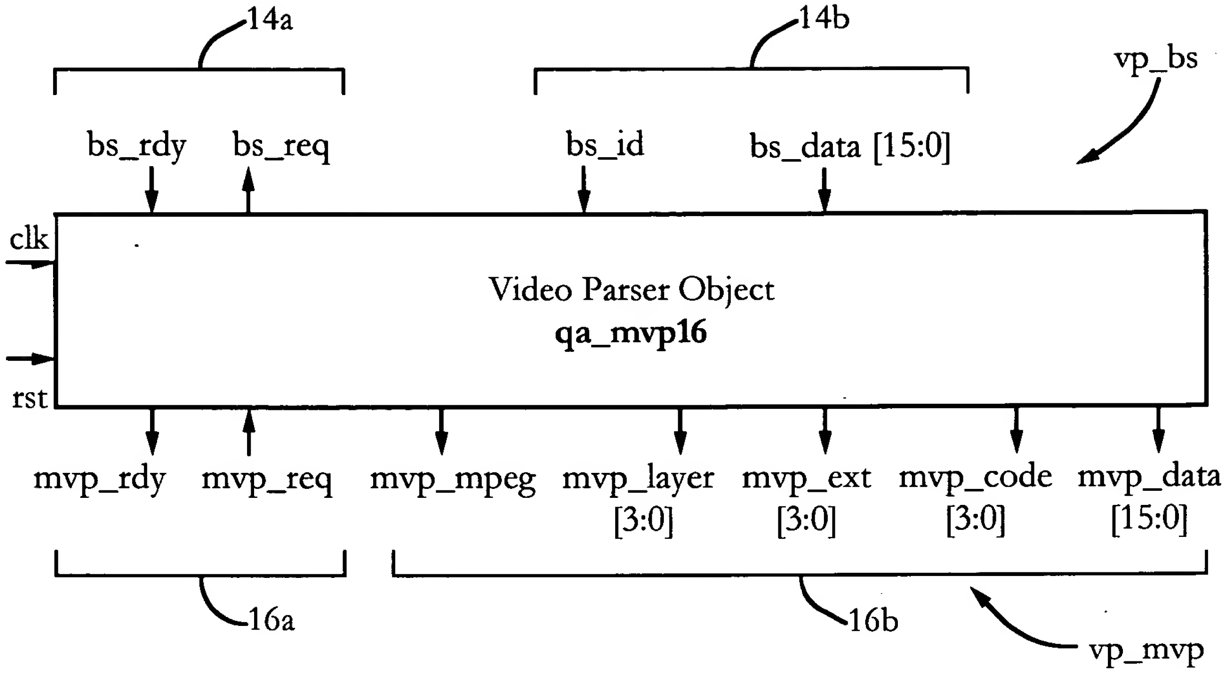
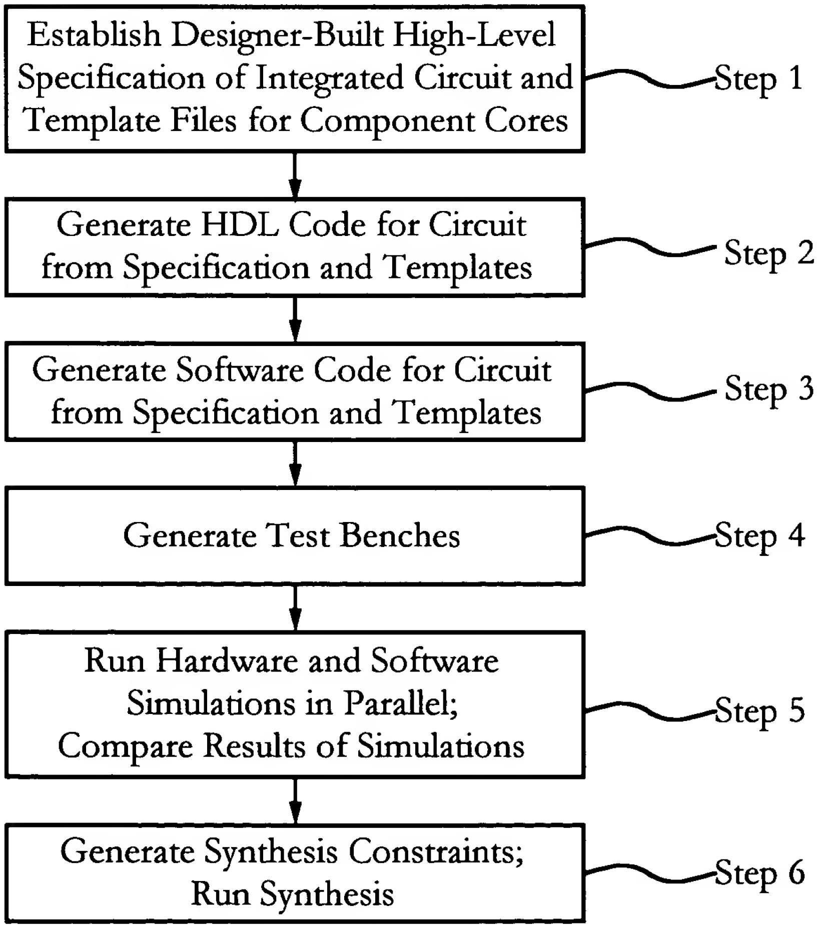

10
qm_m2vd
MPEG-2 Decoder
qa_mvp16
MPEG-2 Video Parser
vp_bs
vp_mvp
13
qm_miq1
mvp
SRAM
qa_miqa
Inverse Quantiser
iqz
SRAM
qa_miqc
Quantiser Control
dat
iq_dat
qa_mmv
MPEG-2 Motion Vectors
mcr_mot
mcw_mot
pat
cmd
qm_idct2
i_dat
qa_idct08
1-D IDCT
c_dat
SRAM
qa_idctc
IDCT Control
r_dat
qa_idct08
1-D IDCT
o_dat
dct_dat
qa_mmc32
MPEG-2 Motion Compensation
mcr_dat
mcw_dat
SRAM
12

FIG. 1



14a
14b
vp_bs
bs_rdy
bs_req
bs_id
bs_data [15:0]
clk
Video Parser Object
qa_mvp16
rst
mvp_rdy
mvp_req
mvp_mpeg
mvp_layer [3:0]
mvp_ext [3:0]
mvp_code [3:0]
mvp_data [15:0]
16a
16b
vp_mvp

FIG. 2



Establish Designer-Built High-Level Specification of Integrated Circuit and Template Files for Component Cores
Step 1
Generate HDL Code for Circuit from Specification and Templates
Step 2
Generate Software Code for Circuit from Specification and Templates
Step 3
Generate Test Benches
Step 4
Run Hardware and Software Simulations in Parallel; Compare Results of Simulations
Step 5
Generate Synthesis Constraints; Run Synthesis
Step 6

FIG. 3-B

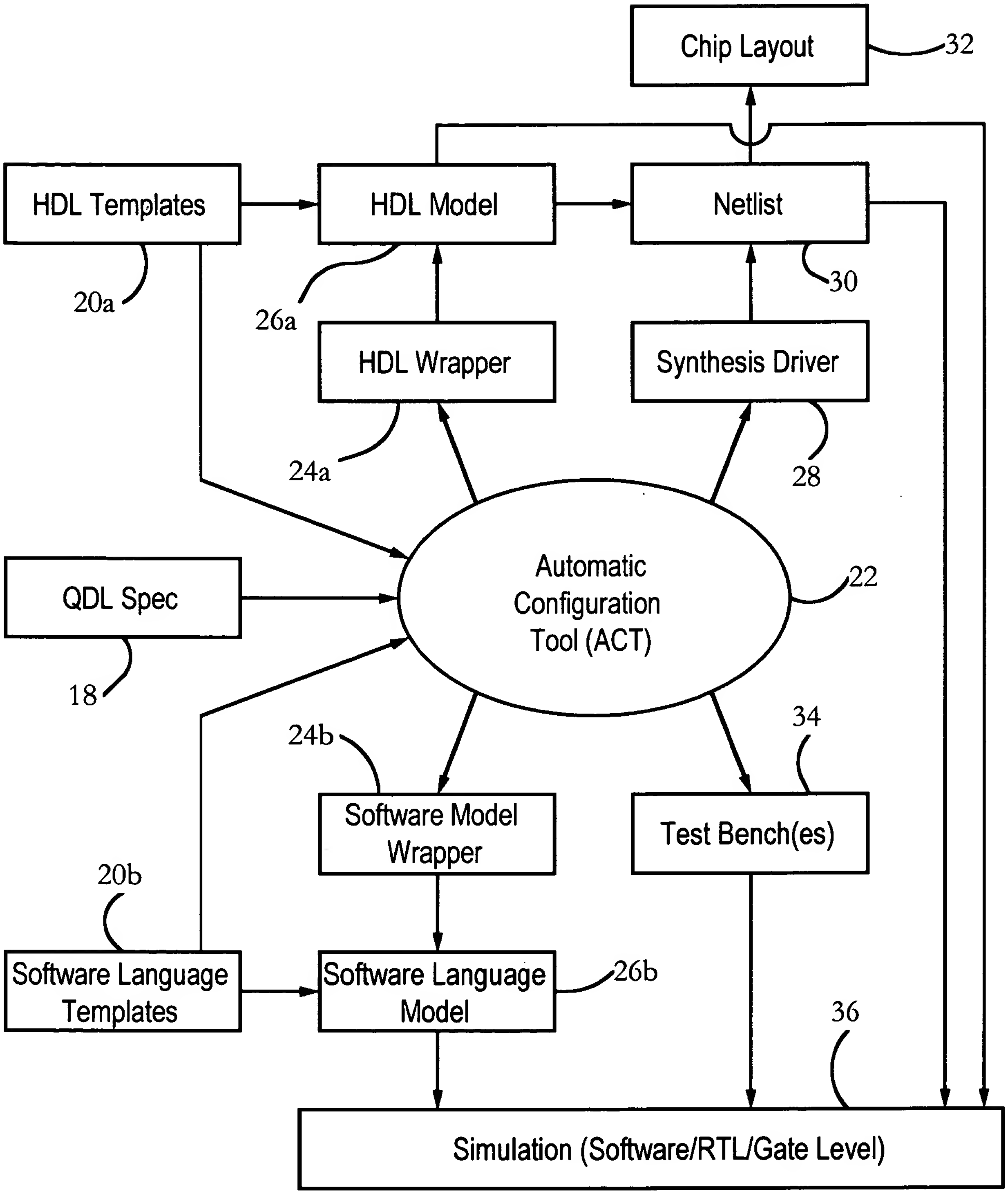

Chip Layout
32
HDL Templates
HDL Model
Netlist
20a
26a
30
HDL Wrapper
Synthesis Driver
24a
28
QDL Spec
Automatic Configuration Tool (ACT)
22
18
24b
34
Software Model Wrapper
Test Bench(es)
20b
Software Language Templates
Software Language Model
26b
36
Simulation (Software/RTL/Gate Level)

FIG. 3-A

```
token bs        {       sig {   field   =  id;}
                        sig {   field   =  data;
                                range   =  [15:0;]}}
token mvp       {       sig {   field   =  mpeg}
                        sig {   field   =  layer;
                                range   =  [3:0];}
                        sig {   field   =  ext;
                                range   =  [3:0];}
                        sig {   field   =  code;
                                range   =  [3:0];}
                        sig {   field   =  data;
                                range   =  [15:0];}}
```

40

FIG. 4

```
object qa_mvp16         {       param BSN  =  'MVP_BSN;
                                param BSW =  LOG2(BSN);
        input           {       port      =  clk;}
        input           {       port      =  rst_;}
        token_in        {       type      =  bs;
                                port      =  vp;}
        token_out       {       type      =  mvp;
                                port      =  vp;}}
```

50

FIG. 5

60b

```
(/* Start: QDL_PORT_LIST */
clk                     ,
rst_                    ,
vp_bs_id                ,
vp_bs_data              ,
vp_bs_rdy               ,
vp_bs_req               ,
vp_mvp_mpeg             ,
vp_mvp_layer            ,
vp_mvp_ext              ,
vp_mvp_code             ,
vp_mvp_data             ,
vp_mvp_rdy              ,
vp_mvp_req
/* End: QDL_PORT_LIST */);
```

```
module   QDL_MOD_NAME
   (/* QDL_PORT_LIST */);
     /* QDL_PARAM_LIST */
     /* QDL_BUS_DEFS */
     /* QDL_PORT_DECL */
     /* QDL_PORT_WIRE */
// Internal Logic Here;
endmodule //QDL_MOD_NAME
```

FIG. 6-A

60a

FIG. 6-B

FIG. 6-C

```
/* Start: QDL_PARAM_LIST */
parameter     BSN     = 'MVP_BSN;
parameter     BSW     =
((BSN)<=2)?1:(((BSN)<=4)?2:(((BSN)<=8)?3:(BSN)<=16)?4:5))));
parameter     MVP_NR = 1
/* End: QDL_PARAM_LIST */
```

60c

FIG. 6-D

```
/* Start: QDL_BUS_DEFS */
parameter       BS_ID_MSB          = (0);
parameter       BS_ID_LSB          = (0);
parameter       BS_ID_W            = (BS_ID_MSB - BS_ID_LSB + 1);
parameter       BS_DATA_MSB        = (16-1);
parameter       BS_DATA_LSB        = (0);
parameter       BS_DATA_W          = (BS_DATA_MSB - BS_DATA_LSB + 1);
parameter       BS_ALL             = BS_ID_W + BS_DATA_W;
parameter       MVP_MPEG_MSB       = (0);
parameter       MVP_MPEG_LSB       = (0);
parameter       MVP_MPEG_W = (MVP_MPEG_MSB - MVP_MPEG_LSB + 1);
//some parameters cut out
parameter       MVP_DATA_MSB       = (15);
parameter       MVP_DATA_LSB       = (0);
parameter       MVP_DATA_W = (MVP_DATA_MSB -MVP_DATA_LSB + 1);
parameter       MVP_ALL = MVP_MPEG_W + MVP_LAYER_W +
                          MVP_EXT_W + MVP_CODE_W + MVP_DATA_W;
/* End: QDL_BUS_DEFS */
```

60d

60g

FIG. 6-G

```
q1_qi #(BS_ALL, 1) qa_mvp16_qi (
          .clk      (clk      ),
          .rst_     (rst_     ),
          .idata    ({vp_bs_id, vp_bs_data}),
          .irdy     (vp_bs_rdy),
          .ireq     (vp _bs_req),
          .odata    ({icmd, idata)},
          .ordy     (irdy),
          .oreq     (ireq),);
```

FIG. 6-E

```
/* Start: QDL_PORT_DECL */
input                                   clk;
input                                   rst_;
input                                   vp_bs_id;
input [BS_DATA_MSB:BS_DATA_LSB]         vp_bs_data;
input [1-1:0]                           vp_bs_rdy;
output [1-1:0]                          vp_bs_req;
output                                  vp_mvp_mpeg;
output [MVP_LAYER_MSB:MVP_LAYER_LSB]    vp_mvp_layer;
output [MVP_EXT_MSB:MVP_EXT_LSB]        vp_mvp_ext;
output [MVP_CODE_MSB:MVP_CODE_LSB]      vp_mvp_code;
output [MVP_DATA_MSB:MVP_DATA_LSB]      vp_mvp_data;
output [MVP_NR-1:0]                     vp_mvp_rdy;
input [MVP_NR-1:0]]                     vp_mvp_req;
/* End: QDL_PORT_DECL */
```

60e

```
/* Start : QDL_PORT_WIRE */
wire [1-1:0]                            vp_bs_req;
wire                                    vp_mvp_mpeg;
wire [MVP_LAYER_MSB:MVP_LAYER_LSB]      vp_mvp_layer;
wire [MVP_EXT_MSB:MVP_EXT_LSB]          vp_mvp_ext;
wire [MVP_CODE_MSB:MVP_CODE_LSB]        vp_mvp_code;
wire [MVP_DATA_MSB:MVP_DATA_LSB         vp_mvp_data;
wire [MVP_NR-1:0]                       vp_mvp_rdy;
/* End : QDL_PORT_WIRE */
```

60f

FIG. 6-F

```
q1_qo #(MVP_ALL, 1, MVP_NR) qa_mvp16_qo (
.clk     (clk      ),
.rst_    (rst_     ),
.idata   ({MpegR, LayerY, ExtR, CodeY, DataY}),
.irdy    (ordy),
.ireq    (oreq),
.odata   ({vp_mvp_mpeg, vp_mvp_layer, vp_mvp_ext,
         vp_mvp_code, vp_mvp_data)},
.ordy    (irdy),
.oreq    (ireq),);
```

60h

FIG. 6-H

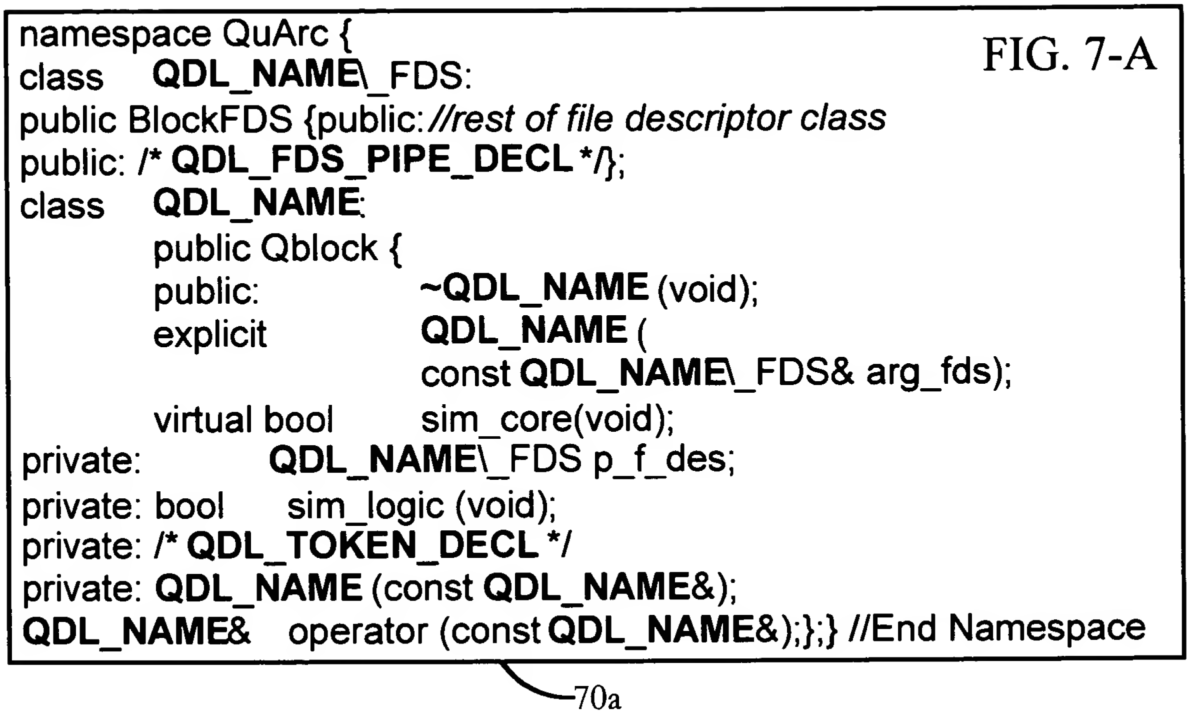


FIG. 7-A

```
namespace QuArc {
class    QDL_NAME\_FDS:
public BlockFDS {public://rest of file descriptor class
public: /* QDL_FDS_PIPE_DECL */};
class    QDL_NAME:
         public Qblock {
         public:              ~QDL_NAME (void);
         explicit             QDL_NAME (
                              const QDL_NAME\_FDS& arg_fds);
         virtual bool         sim_core(void);
private:          QDL_NAME\_FDS p_f_des;
private: bool      sim_logic (void);
private: /* QDL_TOKEN_DECL */
private: QDL_NAME (const QDL_NAME&);
QDL_NAME&    operator (const QDL_NAME&);};} //End Namespace
```

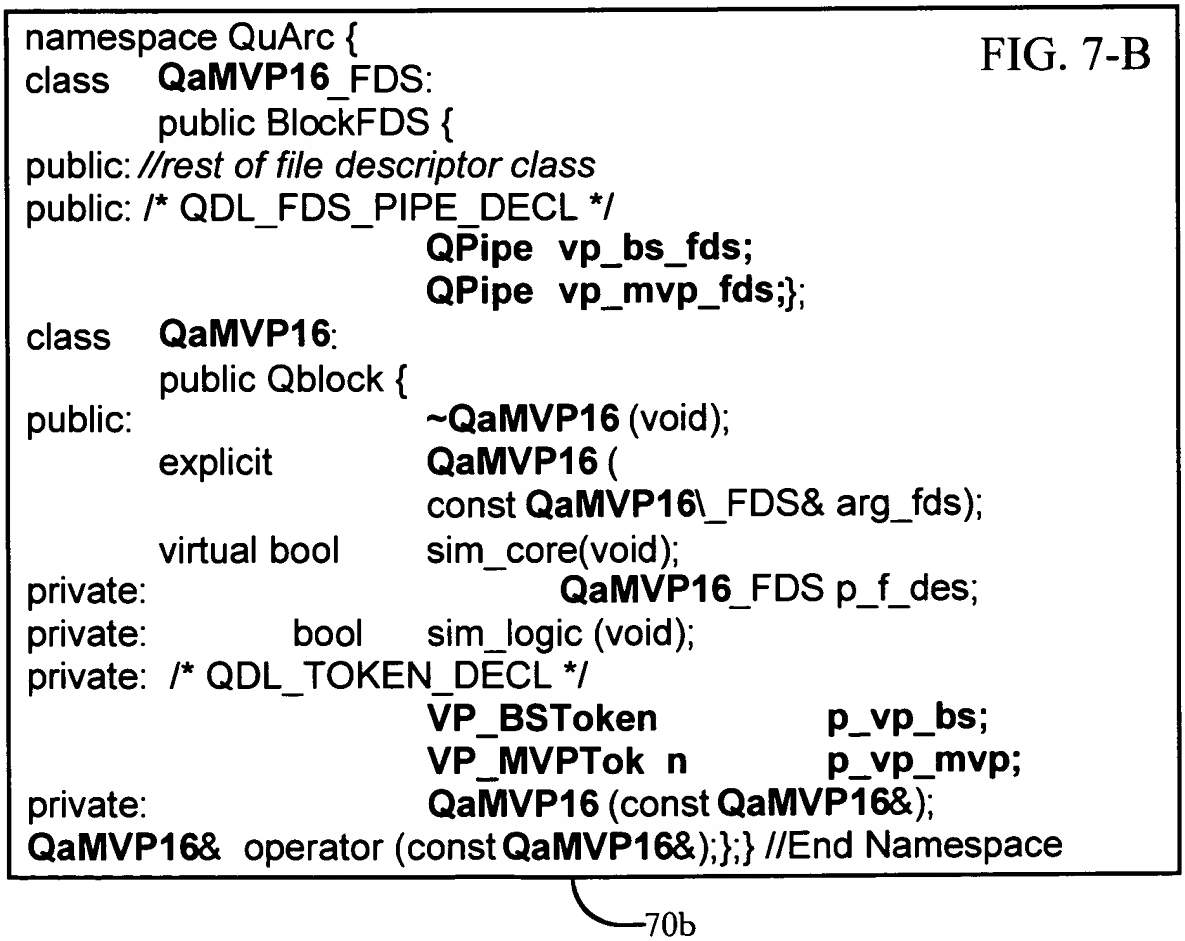


FIG. 7-B

```
namespace QuArc {
class    QaMVP16_FDS:
         public BlockFDS {
public: //rest of file descriptor class
public: /* QDL_FDS_PIPE_DECL */
                              QPipe  vp_bs_fds;
                              QPipe  vp_mvp_fds;};
class    QaMVP16:
         public Qblock {
public:                       ~QaMVP16 (void);
         explicit             QaMVP16 (
                              const QaMVP16\_FDS& arg_fds);
         virtual bool         sim_core(void);
private:                                QaMVP16_FDS p_f_des;
private:          bool        sim_logic (void);
private:  /* QDL_TOKEN_DECL */
                              VP_BSToken          p_vp_bs;
                              VP_MVPTok n         p_vp_mvp;
private:                      QaMVP16 (const QaMVP16&);
QaMVP16&  operator (const QaMVP16&);};} //End Namespace
```

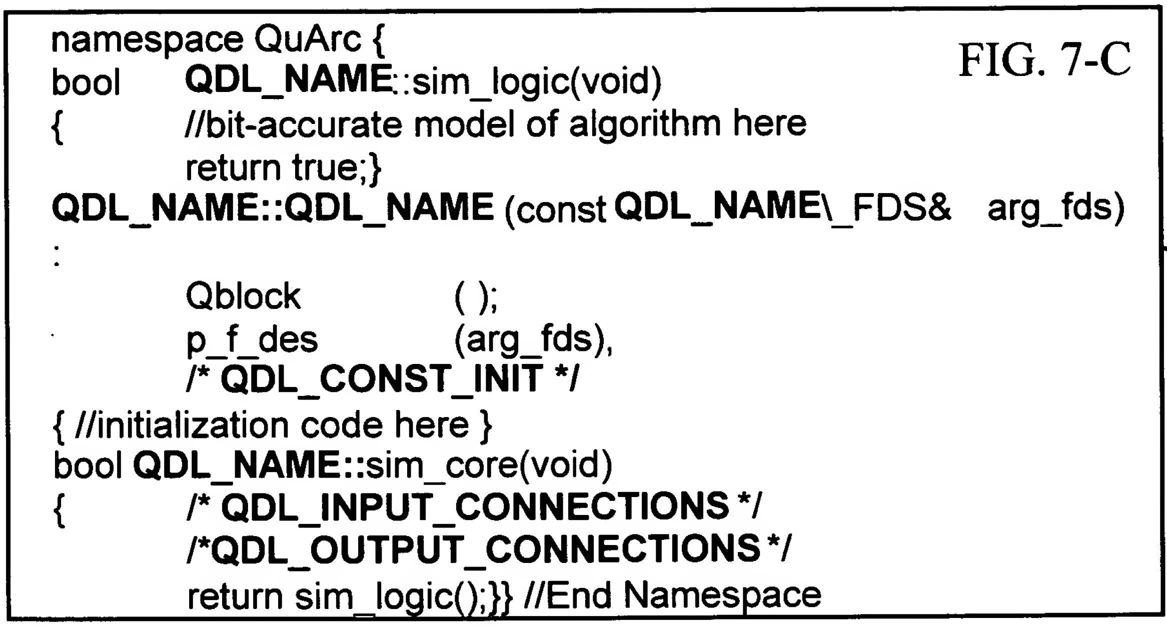


FIG. 7-C

```
namespace QuArc {
bool    QDL_NAME::sim_logic(void)
{       //bit-accurate model of algorithm here
        return true;}
QDL_NAME::QDL_NAME (const QDL_NAME\_FDS&   arg_fds)
:
        Qblock          ( );
        p_f_des         (arg_fds),
        /* QDL_CONST_INIT */
{ //initialization code here }
bool QDL_NAME::sim_core(void)
{       /* QDL_INPUT_CONNECTIONS */
        /*QDL_OUTPUT_CONNECTIONS */
        return sim_logic();}} //End Namespace
```

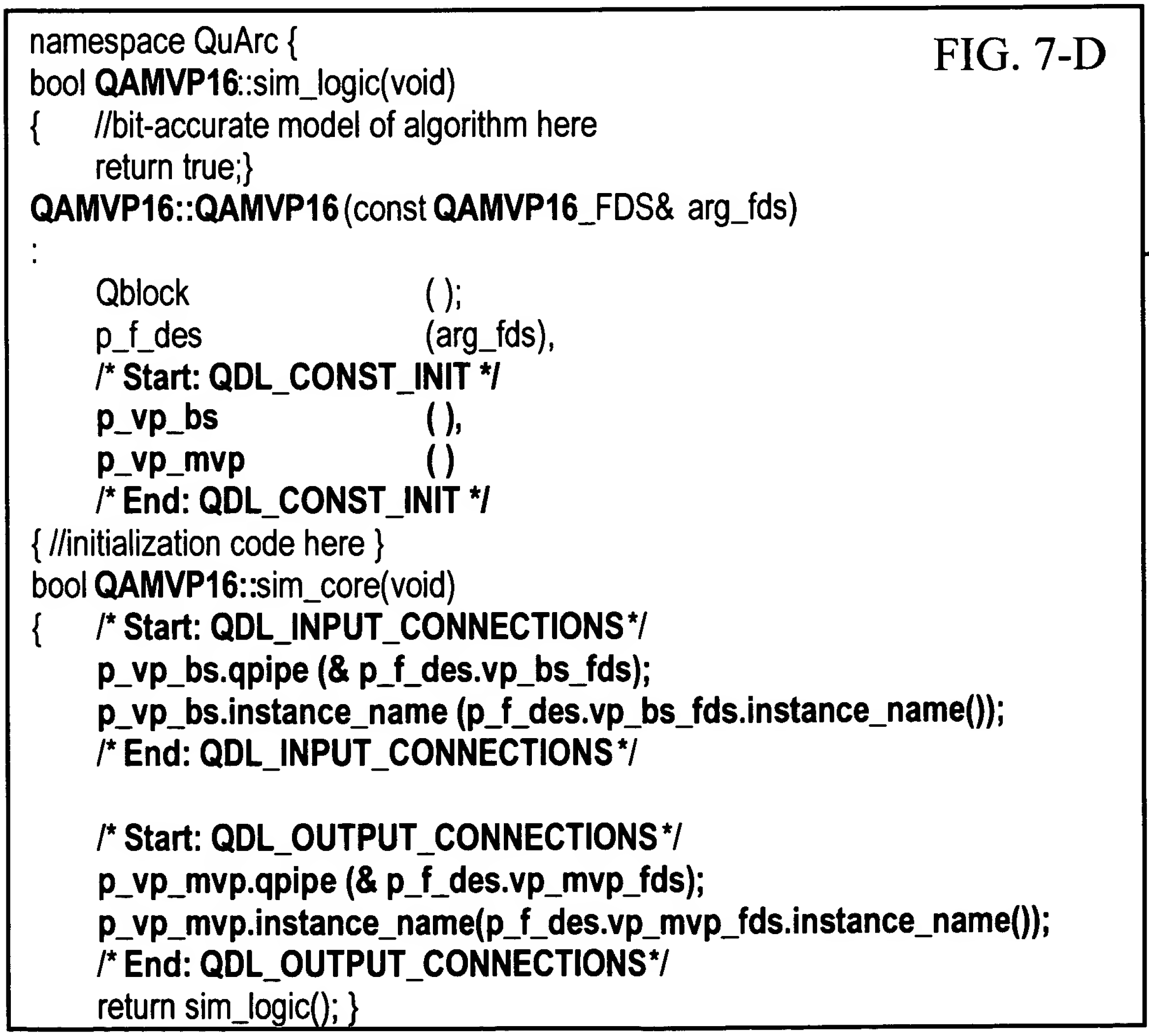


FIG. 7-D

```
namespace QuArc {
bool QAMVP16::sim_logic(void)
{    //bit-accurate model of algorithm here
     return true;}
QAMVP16::QAMVP16(const QAMVP16_FDS& arg_fds)
:
     Qblock            ( );
     p_f_des           (arg_fds),
     /* Start: QDL_CONST_INIT */
     p_vp_bs           ( ),
     p_vp_mvp          ( )
     /* End: QDL_CONST_INIT */
{ //initialization code here }
bool QAMVP16::sim_core(void)
{    /* Start: QDL_INPUT_CONNECTIONS */
     p_vp_bs.qpipe (& p_f_des.vp_bs_fds);
     p_vp_bs.instance_name (p_f_des.vp_bs_fds.instance_name());
     /* End: QDL_INPUT_CONNECTIONS */

     /* Start: QDL_OUTPUT_CONNECTIONS */
     p_vp_mvp.qpipe (& p_f_des.vp_mvp_fds);
     p_vp_mvp.instance_name(p_f_des.vp_mvp_fds.instance_name());
     /* End: QDL_OUTPUT_CONNECTIONS*/
     return sim_logic(); }
```

80a

```
# include ".qdl specs of component Objects"
object qm_miq1          {       param BSN      = 'MIQ1_BSN;
                                param BSW      = LOG2(BSN);
                                REQUIRED PORTS;
                                token_in {
                                        type     = mvp;
                                        port     = vp;}
                                token_out {
                                        type     = dat;
                                        var DW          = 16;}}
```

FIG. 8-A

```
instantiate     {object                     = qa_miqa;
                name                        = iqa;
                param BSN                   = BSN;
                param BSW                   = BSW;
                REQUIRED_CONN;
                connect {       type        = mvp;
                                label       = vp;}
                connect {       type        = iqz;}
                RAM_1port_connect(q, q, 7, 16);}
instantiate     {object                     = qa_miqc;
                name                         = iqc;
                REQUIRED_CONN;
                connect {       type        = iqz;}
                connect {       type        = dat;
                                var DW      = 16;}
                RAM_1r1w_connect(z, z, 6, 12);}}
```

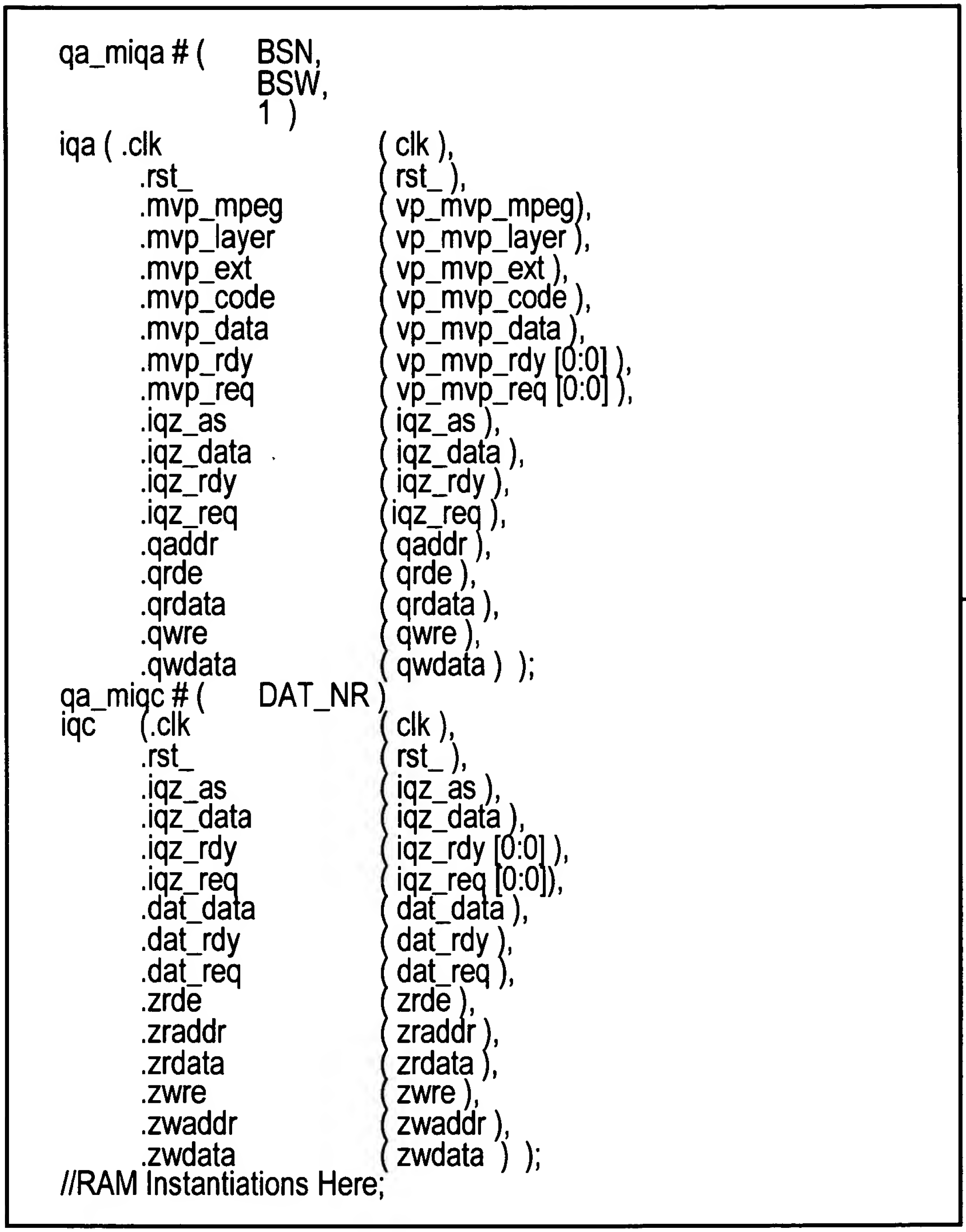
80b

FIG. 8-B

```
module QDL_MOD_NAME
   (/* QDL_PORT_LIST */);
    /* QDL_PARAM_LIST */
    /* QDL_BUS_DEFS */
    /* QDL_PORT_DECL */
    /* QDL_PORT_WIRE */
    /* QDL_INSTANCE */
endmodule // QDL_MOD_NAME
```

90a

FIG. 9-A



```
qa_miqa # (     BSN,
                BSW,
                1  )
iqa ( .clk                  ( clk ),
        .rst_               ( rst_ ),
        .mvp_mpeg           ( vp_mvp_mpeg),
        .mvp_layer          ( vp_mvp_layer ),
        .mvp_ext            ( vp_mvp_ext ),
        .mvp_code           ( vp_mvp_code ),
        .mvp_data           ( vp_mvp_data ),
        .mvp_rdy            ( vp_mvp_rdy [0:0] ),
        .mvp_req            ( vp_mvp_req [0:0] ),
        .iqz_as             ( iqz_as ),
        .iqz_data           ( iqz_data ),
        .iqz_rdy            ( iqz_rdy ),
        .iqz_req            (iqz_req ),
        .qaddr              ( qaddr ),
        .qrde               ( qrde ),
        .qrdata             ( qrdata ),
        .qwre               ( qwre ),
        .qwdata             ( qwdata )  );
qa_miqc # (     DAT_NR )
iqc     (.clk               ( clk ),
        .rst_               ( rst_ ),
        .iqz_as             ( iqz_as ),
        .iqz_data           ( iqz_data ),
        .iqz_rdy            ( iqz_rdy [0:0] ),
        .iqz_req            ( iqz_req [0:0]),
        .dat_data           ( dat_data ),
        .dat_rdy            ( dat_rdy ),
        .dat_req            ( dat_req ),
        .zrde               ( zrde ),
        .zraddr             ( zraddr ),
        .zrdata             ( zrdata ),
        .zwre               ( zwre ),
        .zwaddr             ( zwaddr ),
        .zwdata             ( zwdata  )  );
//RAM Instantiations Here;
```

FIG. 9-B

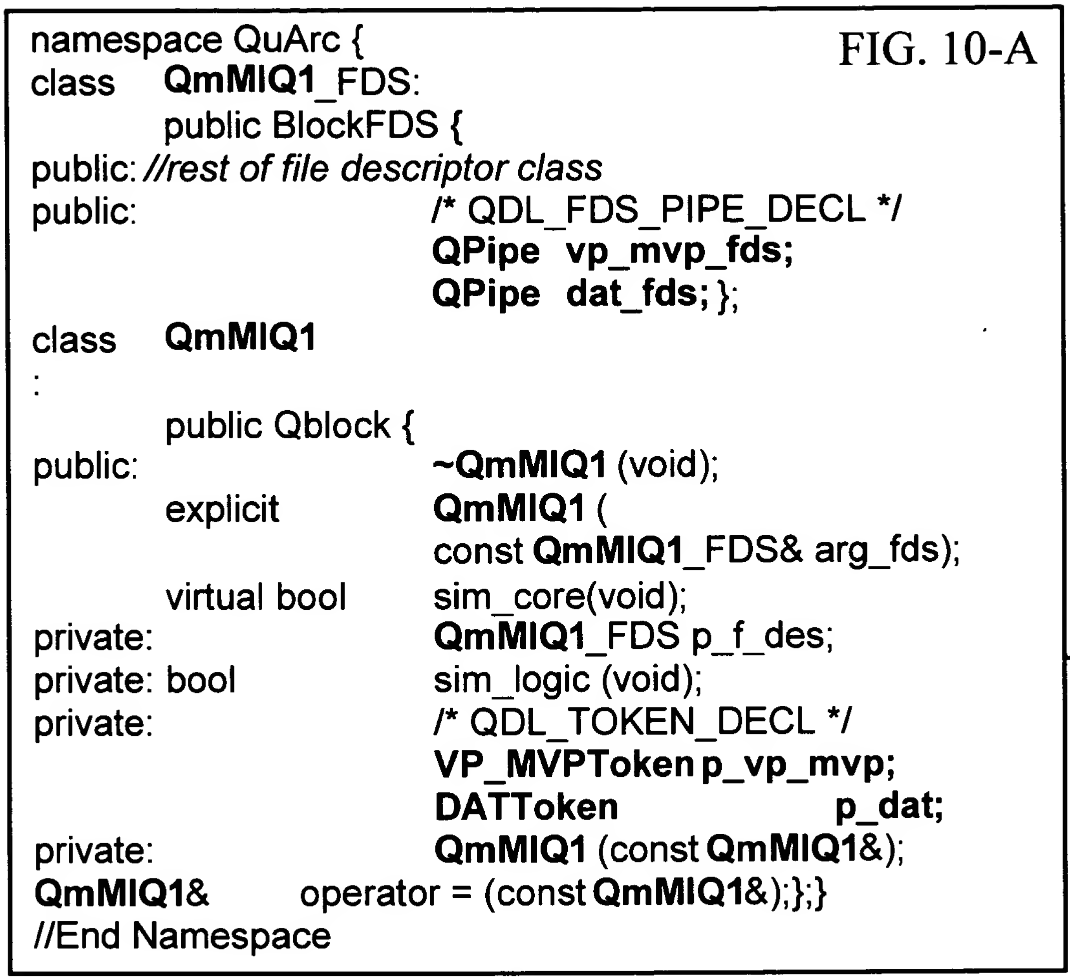


FIG. 10-A

```
namespace QuArc {
class    QmMIQ1_FDS:
         public BlockFDS {
public: //rest of file descriptor class
public:                  /* QDL_FDS_PIPE_DECL */
                         QPipe  vp_mvp_fds;
                         QPipe  dat_fds; };
class    QmMIQ1
:
         public Qblock {
public:                  ~QmMIQ1 (void);
         explicit        QmMIQ1 (
                         const QmMIQ1_FDS& arg_fds);
         virtual bool    sim_core(void);
private:                 QmMIQ1_FDS p_f_des;
private: bool            sim_logic (void);
private:                 /* QDL_TOKEN_DECL */
                         VP_MVPToken p_vp_mvp;
                         DATToken            p_dat;
private:                 QmMIQ1 (const QmMIQ1&);
QmMIQ1&      operator = (const QmMIQ1&);};}
//End Namespace
```

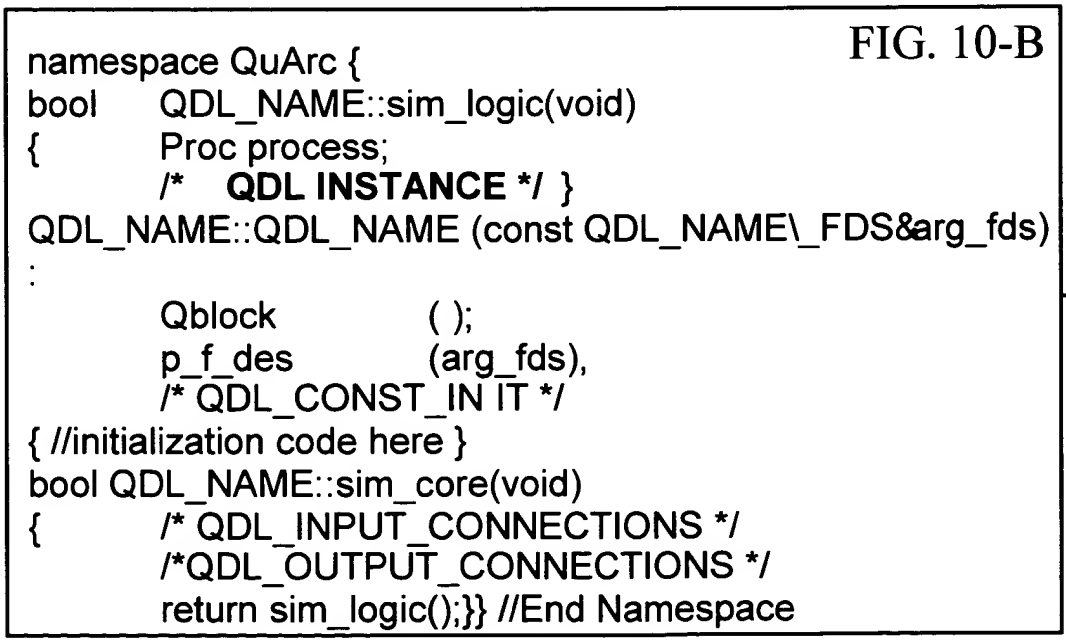


FIG. 10-B

```
namespace QuArc {
bool     QDL_NAME::sim_logic(void)
{        Proc process;
         /*   QDL INSTANCE */ }
QDL_NAME::QDL_NAME (const QDL_NAME\_FDS&arg_fds)
:
         Qblock          ( );
         p_f_des         (arg_fds),
         /* QDL_CONST_IN IT */
{ //initialization code here }
bool QDL_NAME::sim_core(void)
{        /* QDL_INPUT_CONNECTIONS */
         /*QDL_OUTPUT_CONNECTIONS */
         return sim_logic();}} //End Namespace
```

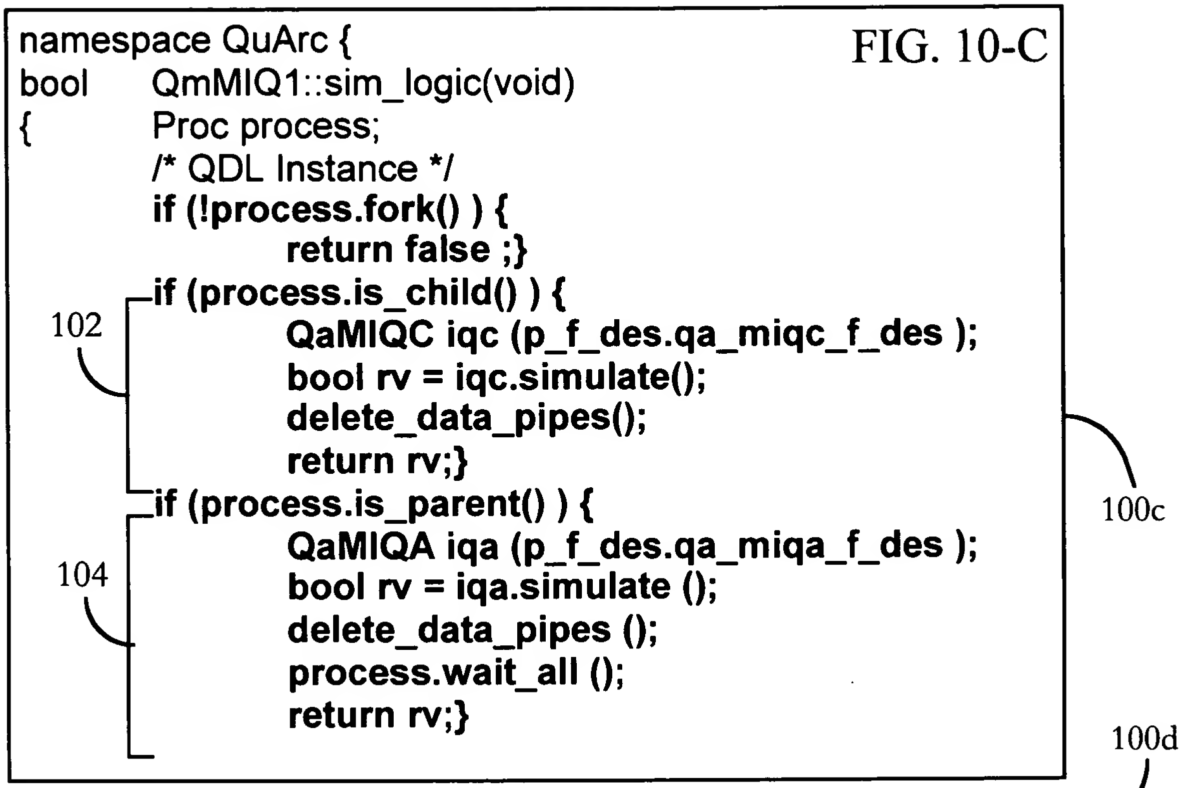
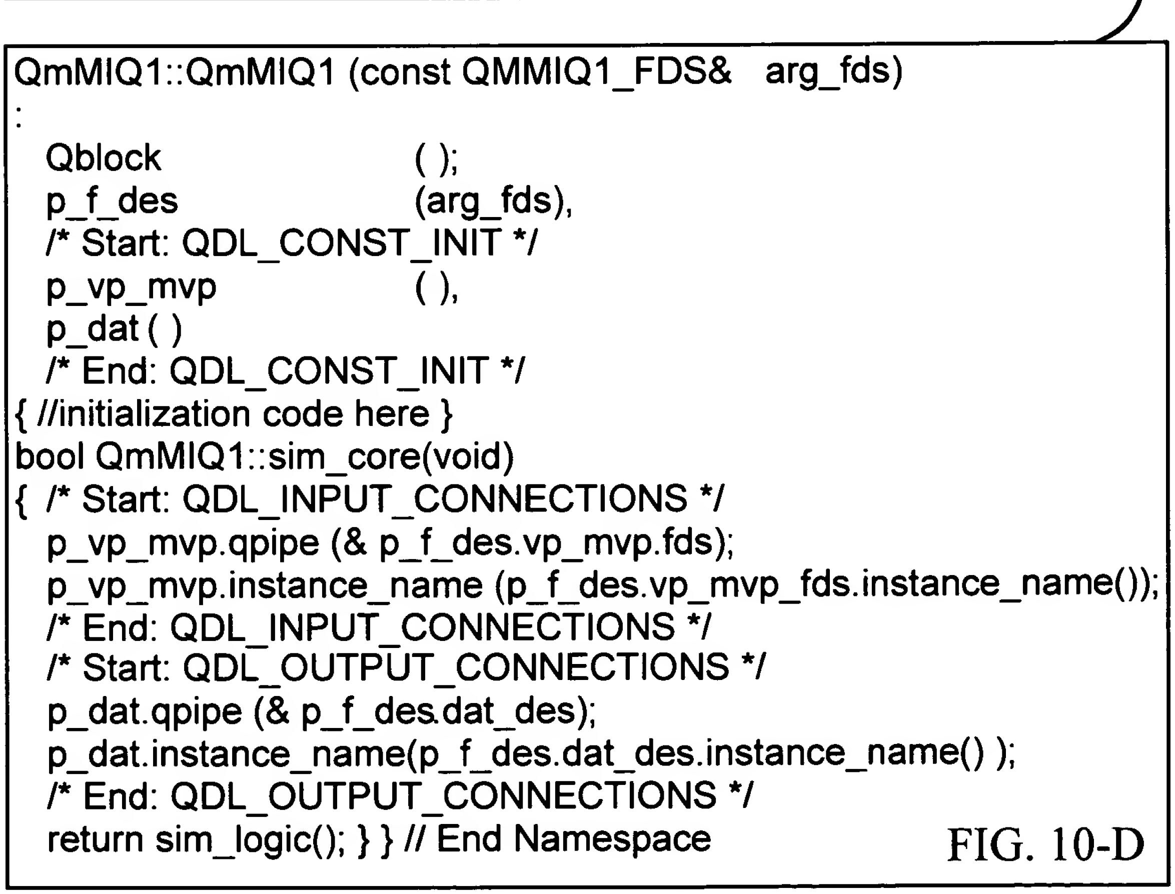


FIG. 10-C

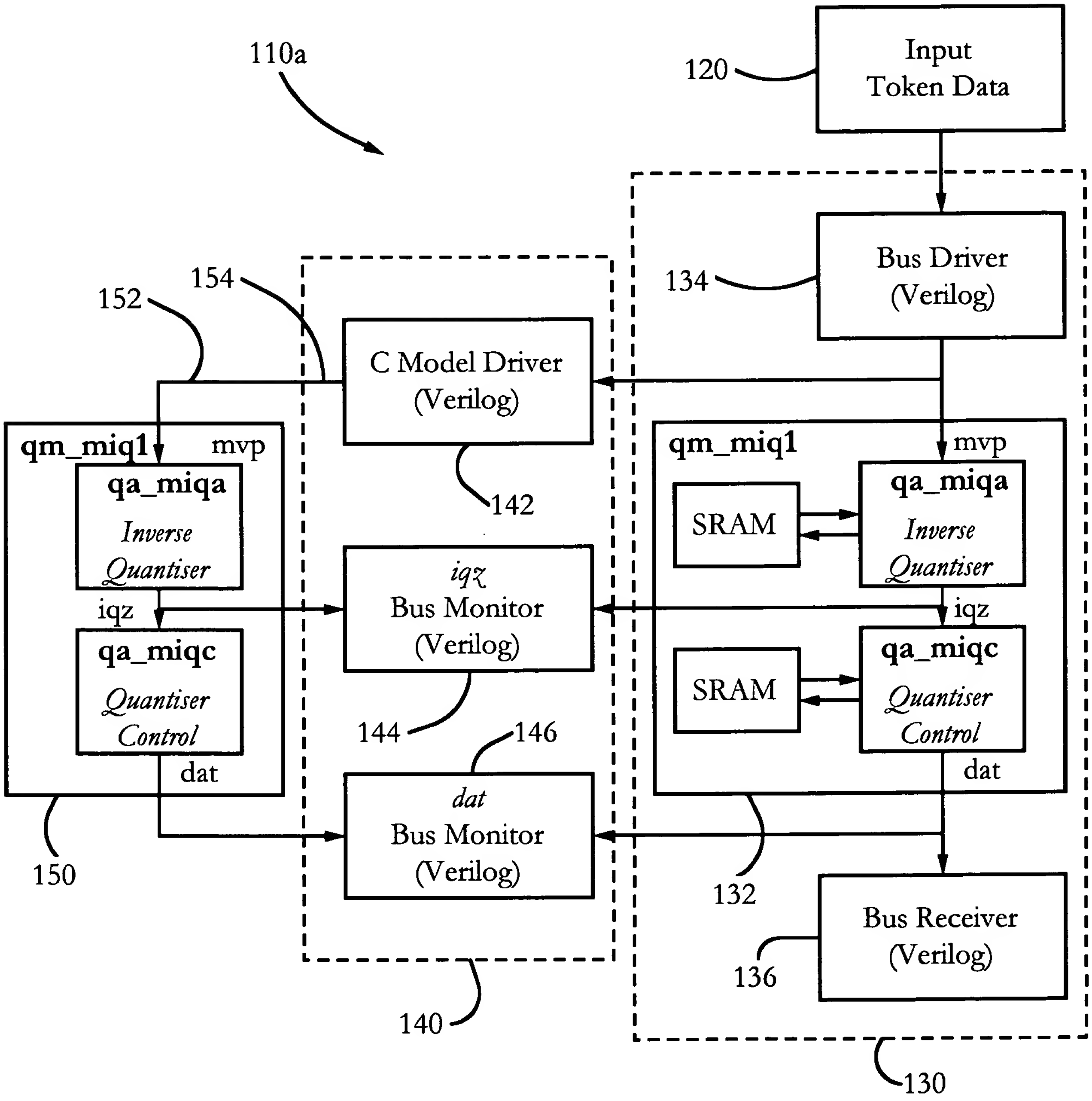
```
namespace QuArc {
bool      QmMIQ1::sim_logic(void)
{         Proc process;
          /* QDL Instance */
          if (!process.fork() ) {
                    return false ;}
          if (process.is_child() ) {
                    QaMIQC iqc (p_f_des.qa_miqc_f_des );
                    bool rv = iqc.simulate();
                    delete_data_pipes();
                    return rv;}
          if (process.is_parent() ) {
                    QaMIQA iqa (p_f_des.qa_miqa_f_des );
                    bool rv = iqa.simulate ();
                    delete_data_pipes ();
                    process.wait_all ();
                    return rv;}
```

102
104
100c



```
QmMIQ1::QmMIQ1 (const QMMIQ1_FDS&   arg_fds)
:
  Qblock                    ( );
  p_f_des                   (arg_fds),
  /* Start: QDL_CONST_INIT */
  p_vp_mvp                  ( ),
  p_dat ( )
  /* End: QDL_CONST_INIT */
{ //initialization code here }
bool QmMIQ1::sim_core(void)
{ /* Start: QDL_INPUT_CONNECTIONS */
  p_vp_mvp.qpipe (& p_f_des.vp_mvp.fds);
  p_vp_mvp.instance_name (p_f_des.vp_mvp_fds.instance_name());
  /* End: QDL_INPUT_CONNECTIONS */
  /* Start: QDL_OUTPUT_CONNECTIONS */
  p_dat.qpipe (& p_f_des.dat_des);
  p_dat.instance_name(p_f_des.dat_des.instance_name() );
  /* End: QDL_OUTPUT_CONNECTIONS */
  return sim_logic(); } } // End Namespace
```

FIG. 10-D



110a
120
Input Token Data
134
Bus Driver (Verilog)
152
154
C Model Driver (Verilog)
142
qm_miq1
mvp
qa_miqa
Inverse Quantiser
iqz
qa_miqc
Quantiser Control
dat
150
iqz Bus Monitor (Verilog)
144
146
dat Bus Monitor (Verilog)
140
qm_miq1
SRAM
qa_miqa
Inverse Quantiser
iqz
SRAM
qa_miqc
Quantiser Control
dat
132
Bus Receiver (Verilog)
136
130

FIG. 11-A

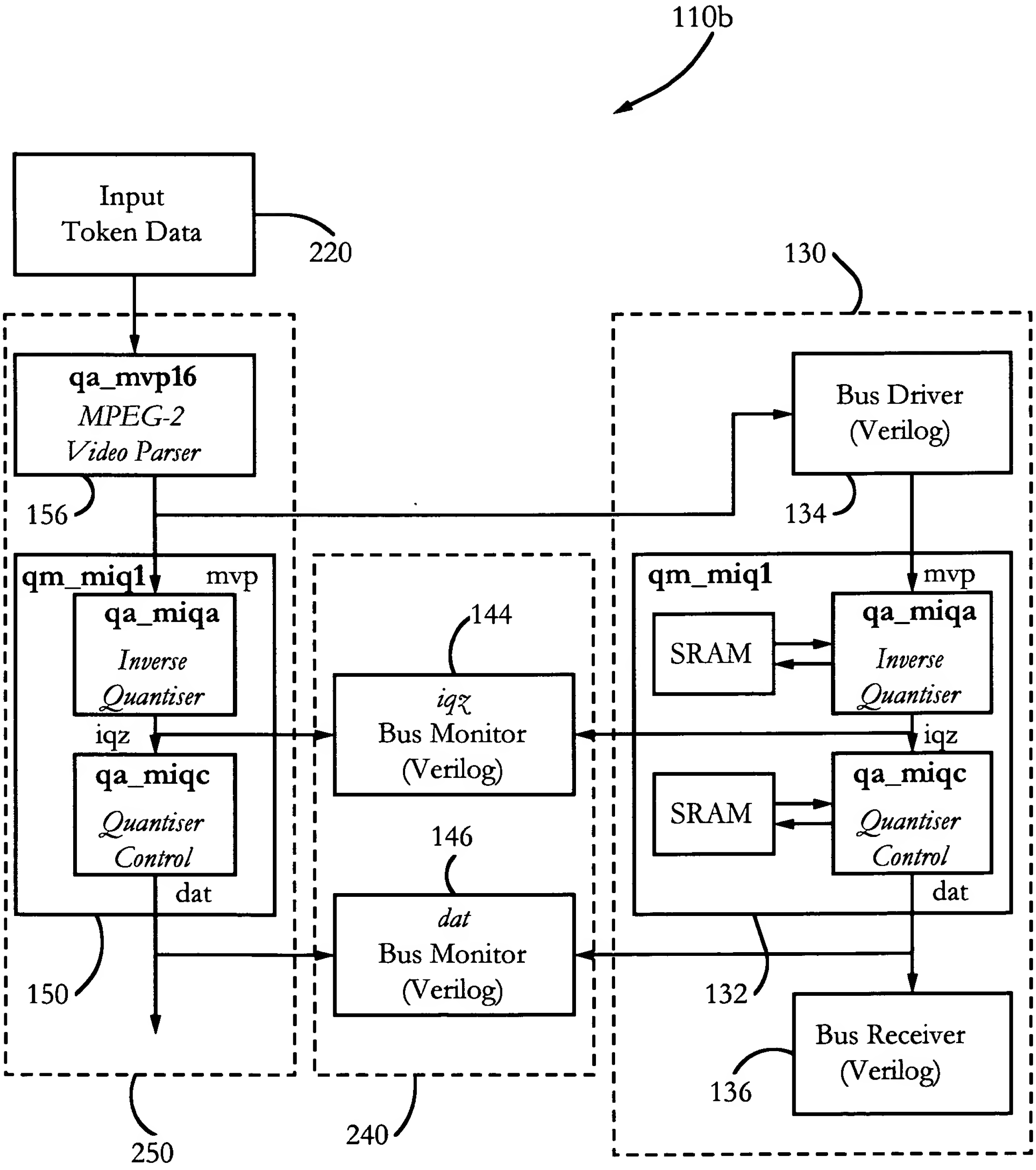

110b
Input
Token Data
220
130
qa_mvp16
MPEG-2
Video Parser
156
Bus Driver
(Verilog)
134
qm_miq1
mvp
qa_miqa
Inverse
Quantiser
iqz
qa_miqc
Quantiser
Control
dat
150
144
iqz
Bus Monitor
(Verilog)
146
dat
Bus Monitor
(Verilog)
240
250
qm_miq1
SRAM
qa_miqa
Inverse
Quantiser
iqz
SRAM
qa_miqc
Quantiser
Control
dat
132
Bus Receiver
(Verilog)
136

FIG. 11-B

```
setglobal    if_input_delay        [expr 0.75 * $cycle_time - $clock_skew]
setglobal    if_output_delay       [expr 0.75 * $cycle_time - $clock_skew]
setglobal    token_input_delay     [expr 0.85 * $cycle_time - $clock_skew]
setglobal    token_output_delay    [expr 0.85 * $cycle_time - $clock_skew]
setglobal    ram_input_delay       [expr 0.75 * $cycle_time - $clock_skew]
setglobal    ram_output_delay      [expr 0.25 * $cycle_time - $clock_skew]
setglobal    token_crit_range      [expr 0.10 * $cycle_time]
setglobal    ram_crit_range        [expr 0.10 * $cycle_time]
setglobal    def_crit_range        [expr 0.10 * $cycle_time]
setglobal    clk_crit_range        [expr 0.10 * $cycle_time]
setglobal    def_load              [expr 4 * [load_of “lib. inverter cell”]]
setglobal    qif_load              [expr 5 * $def_load]
setglobal    def_drive             [drive_of “library NAND2 cell”]
```

320a

FIG. 12-A

```
proc    QsynSetTokenInConstraint { bus } {
        upvar token_input_delay token_output_delay ...(rest of variables)
        set token_ports     [get_ports “$(bus)*”]
        set in_ports        [filter $token_ports “@port_direction == in”]
        set out_ports       [filter $token_ports “@port_direction == out”]
        set_input_delay $token_input_delay -clock $clock $in_ports
        set_output_delay $token_output_delay -clock $clock $out_ports
        set_input_delay $if_input_delay -clock $clock $(bus)rdy
        set_output_delay $if_output_delay -clock $clock $(bus)req
        set_load $qif_load $out_ports
        set_drive $def_drive $in_ports
        group_path -critical_range $token_crit_range -name $bus -to $out_ports
        unset token_ports
        unset in_ports
        unset out_ports
        return 1}
```

320b

FIG. 12-B